Hybridation of Bayesian networks and evolutionary algorithms for multi-objective optimization in an integrated product design and project management context

Paul Pitiot, Thierry Coudert, Laurent Geneste, Claude Baron

Hybridation of Bayesian networks and evolutionary algorithms for multi-objective optimization in an integrated product design and project management context

Paul Pitiot [a,c], Thierry Coudert [a,*], Laurent Geneste [a], Claude Baron [b]

[a] Laboratoire Génie de Production, Ecole Nationale d'Ingénieurs de Tarbes, 47 av. d'Azereix, BP 1629, 65016 Tarbes, France
[b] Laboratoire Toulousain de Technologie et d'Ingénierie des Systèmes, INSA de Toulouse, 135 av. de Rangueil, 31077 Toulouse, France
[c] Centre de Génie Industriel, Ecole des Mines d'Albi, Université de Toulouse Campus Jarlard, 81013 Albi CT Cedex 09, France

A B S T R A C T

A better integration of preliminary product design and project management processes at early steps of system design is nowadays a key industrial issue. Therefore, the aim is to make firms evolve from classical sequential approach (first product design the project design and management) to new integrated approaches. In this paper, a model for integrated product/project optimization is first proposed which allows taking into account simultaneously decisions coming from the product and project managers. However, the resulting model has an important underlying complexity, and a multi-objective optimization technique is required to provide managers with appropriate scenarios in a reasonable amount of time. The proposed approach is based on an original evolutionary algorithm called evolutionary algorithm oriented by knowledge (EAOK). This algorithm is based on the interaction between an adapted evolutionary algorithm and a model of knowledge (MoK) used for giving relevant orientations during the search process. The evolutionary operators of the EA are modified in order to take into account these orientations. The MoK is based on the Bayesian Network formalism and is built both from expert knowledge and from individuals generated by the EA. A learning process permits to update probabilities of the BN from a set of selected individuals. At each cycle of the EA, probabilities contained into the MoK are used to give some bias to the new evolutionary operators. This method ensures both a faster and effective optimization, but it also provides the decision maker with a graphic and interactive model of knowledge linked to the studied project. An experimental platform has been developed to experiment the algorithm and a large campaign of tests permits to compare different strategies as well as the benefits of this novel approach in comparison with a classical EA.

Keywords:
Project management
Product preliminary design
Evolutionary algorithm
Experience feedback
Bayesian network
Learning

1. Introduction

Many companies, in order to meet the requirements of their clients and to provide them with adequate products, implement two key processes:

- the "product design" process, which aims at defining precisely the components and the structure of the product,
- the "project design" process which aims at specifying how the product will be realized (sequence of tasks, used resources…).

These two processes are often implemented sequentially: first the product is designed then the realization project is elaborated. For example, when a client wants to build a house, the architect designs at first a plan of the house, then the corresponding realization project is developed and launched. Since the project constraints (for example resources availability) are not explicitly taken into account in the product design, this can lead to additional iterations between "product design" and "project design" processes. A better integration (or coupling) of both processes is therefore a way to improve the global performance of companies.

An in-depth study of several mechanisms that can facilitate this integration has been launched in a project called ATLAS, funded by the French National Research Agency and involving academic laboratories, industrialists and the competitiveness cluster Aerospace Valley. The work presented in this paper takes place in the context of the ATLAS project.

In this section, a simplified product/project integration model is proposed. Indeed, in both environments (product and project), design processes are generally achieved according to a

Corresponding author.
E-mail addresses: paul.pitiot@enit.fr (P. Pitiot), thierry.coudert@enit.fr (T. Coudert), laurent.geneste@enit.fr (L. Geneste), claude.baron@insa-toulouse.fr (C. Baron).

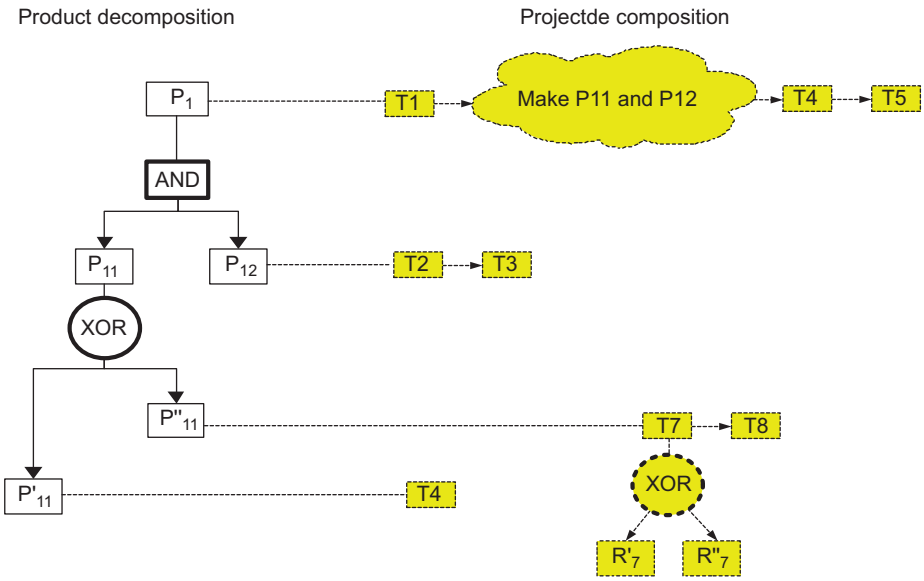


Fig. 1. Example of product/project decomposition.

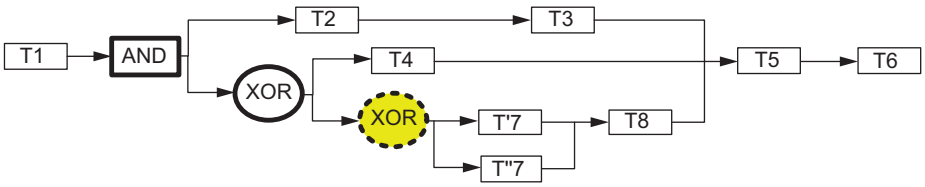


Fig. 2. Example of integrated model: the project graph.

hierarchical decomposition (see Fig. 1) in order to encompass complexity:

- products are recursively decomposed into smaller sub-products ("AND" connectors), e.g. product P_1 is made of P_{11} and P_{12},
- accordingly, projects are recursively decomposed into sub-projects: in order to realize P_1 one has to realize P_{11}, to realize P_{12} and to assemble P_{11} and P_{12},
- alternatives ("XOR" connectors) can be defined in products (e.g. choice between components P_{11} is composed of P'_{11} XOR P''_{12}) and in projects (e.g. choice between sub-contractors R'_7 XOR R''_7 to achieve task T_7).

Definition 1. an integrated model, called **project graph**, is used in order to represent simultaneously the links between the product and project hierarchies. This model consists in an oriented graph without cycles in which nodes are: tasks of the project, "AND" connectors and "XOR" connectors. The oriented arcs represent the precedence constraints between tasks. Fig. 2 represents such a model for the example of decomposition given in Fig. 1. Such a graph permits to capitalize that is called "structural knowledge" in the rest of the article. It concerns XOR nodes that correspond to the possible choices of products' structure. Making a product choice corresponding to a XOR node imply to inhibit a set of downstream connected nodes. Those product XOR are represented by a circular node whereas project XOR, which do not involve inhibition of other XOR node, are represented by dotted circle.

Definition 2. a **scenario** corresponds to a graph in which all the choices are made (i.e. with no more XOR nodes). An example of scenario, corresponding to the model in Fig. 2, is illustrated in Fig. 3.

1.1. Mathematical description of the addressed problem:

The problem addressed in this paper consists in searching an optimal scenario among all the possible ones within the simple directed graph project. The project graph $G=(\alpha, \beta)$ is defined by:

$\alpha=\{\alpha_k\}$, the set of all nodes,
$\beta=\{\beta_{ij}\}$, the set of directed edges between the node α_i and the node α_j with $|\beta|$ the total number of edges.

The following subsets permit to formalize the problem defining the three different node types (XOR, AND, Task):

$T=\{\alpha_p/\alpha_p$ is a task node, $\alpha_p \in \alpha\}$ is the subset of task nodes with $|T|$ the total number of task nodes,
$XOR=\{\alpha_q/\alpha_q$ is a XOR node, $\alpha_q \in \alpha\}$ is the subset of XOR nodes with $|XOR|$ the total number of XOR nodes,
$AND=\{\alpha_r/\alpha_r$ is an AND node, $\alpha_r \in \alpha\}$ is the subset of AND nodes with $|AND|$ the total number of AND nodes.

Let X, the vector of discrete variables (decision variables) corresponding to XOR nodes

$$X = \{x_i/\alpha_i \in XOR\} \quad (1)$$

Let Dx_i, the domain of the variable x_i defined by the vector of identifiers of the direct successor nodes of the XOR node i.

$$Dx_i = \{j/\alpha_j \in \alpha, \beta_{ij} \in \beta, \ \alpha_i \in XOR\} \quad (2)$$

A decision associated to a XOR node α_k that participates to a scenario s corresponds to the instantiation of the variable x_k and is defined by

$$x_k^s = j \quad \text{with} \quad j \in Dx_k \quad (3)$$

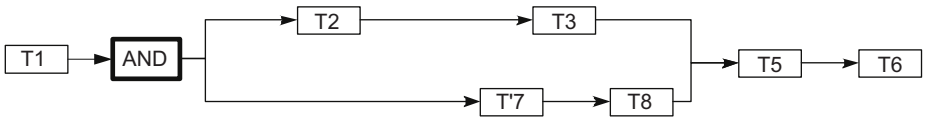


Fig. 3. Example of scenario *s*.

Let X_s, the vector of instantiations of all the variables corresponding to the scenario *s*

$$X_s = \{x_k^s / \alpha_k \in XOR\} \quad (4)$$

Let $G_s(X_s)$ the directed graph obtained by instantiation of all the variables and corresponding to the scenario *s* (e.g. see Fig. 2). $G_s(X_s)=(\alpha_s, \beta_s)$. α_s is the set of nodes belonging to the scenario s and β_s, the set of edges. G_s depends on the variables instantiation X_s.

Let $f_m(x_s)$, the value of the criteria m for the scenario s that depends on the variables instantiation X_s. This value is computed from the graph $G_s(X_s)$ and depends on the considered criteria m. For instance, if the criteria m is the cost, $f_m(X_s)$ is equal to the sum of all the elementary task costs. If the criteria m is the delay, it is necessary to find the longest path in the graph G_s and then, $f_m(X_s)$ represents the final delay of the scenario. Therefore, considering a scenario s, each criteria m is evaluated in a specific manner using an appropriate algorithm to apply to the graph G_s.

Let $f_m(X)$, the objective function to optimize corresponding to the criteria m and depending on the variables X. Considering that there is P objective functions to optimize, the multi-valuated optimization function f is defined by

$$\min f(X) = \min(f_1(X), \quad f_2(X), \ldots, f_P(X)) \quad (5)$$

This problem can be considered as an extended product configuration optimization problem. The existing literature on the subject is mainly dedicated to finding a feasible configuration according to constraints and knowledge on the domain. However, as mentioned in Li et al. (2006) it is very difficult to optimize the resulting configured product since a problem of combinatorial explosion appears especially when the problem is loosely constrained. In this case, using an optimization approach can help to focus on good solutions. In Baron et al. (2004) a search method, based on a classical multi-objective evolutionary algorithm, was proposed for the problem of scenario selection with promising results.

The method proposed in Baron et al. (2004) is improved by taking into account the knowledge that can be capitalized from previous optimizations (learning from experience). Another important issue is to make the capitalized knowledge explicitly available to the decision maker and, therefore, to help him understand the proposed solutions. This enables to avoid the black-box effect of a combined simulation-optimization approach without knowledge acquisition. The main idea developed in this paper is to guide the search process with the available knowledge and, reciprocally, to improve the knowledge by learning from the most interesting solutions obtained during search.

The background of this work, with respect to existing approaches that mix learning and search, is given in Section 2. Then, the proposed approach, based on a hybridization between bayesian networks (for knowledge acquisition) and evolutionary algorithms (for search) is described in Section 3. Finally, the results obtained on the target problem are discussed in Section 4.

2. Background

The method proposed in this paper relates to a recent family of algorithms called "intelligent evolutionary optimization" (IEO) (Huyet and Paris, 2004, Michalski et al., 2006). As stated above, these algorithms are based on the interaction between a search process and a knowledge acquisition process achieved through learning. The goal is to benefit of the advantages of both approaches. The search process aims at improving a set of solutions by pseudo-random selection and combination operations. The goal of the learning process is to extract, to capitalize some knowledge contained into the solutions in order to guide the search process. Indeed, Michalski et al. (2006) shows that fixing some interesting solutions properties is generally enough for the search method to focus very quickly on some solutions close to the optimal. So the learning process has only to give some orientations to the search process with respect to a given context. This section presents works which relate to the scenario selection problem for each process (search or learning optimization methods) then an overview of existing hybrid approaches.

2.1. Search process

The model defined in the previous section represents a highly combinatorial (multiple XOR disjunctions in the graph) multi-objective problem. The multi-objective aspect invites to provide the decision maker with a panel of good solutions which represents various compromises between identified objectives (Pareto front). This kind of problem is often addressed using metaheuristic optimization methods (see Rochet and Baron, 2006 and Chelouah et al., 2009) for studies of different metaheuristic applied to the scenarios selection problem). Among those, this study relies on evolutionary algorithms (EA) (Holland, 1975), as illustrated on the left part of Fig. 4. EA is indeed well suited for multi-criteria optimization and can provide the learning algorithm with a set of individuals that "represent" the global search space.

This kind of method indirectly reuses knowledge related to the problem via the evaluation of the generated solutions. Here, knowledge on the problem corresponds to a connection between a given instantiation of genes (a scheme) and an interesting area of the objective space. Holland (1975) showed that the improvement of individuals in the evolutionary algorithms is ensured by the indirect selection of the schemes with good performance (according to the schemata theorem). Nevertheless in classical EA, unguided evolutionary operators handle genes in a random way. Some techniques try to identify and preserve combinations with good performances thanks to specific search operators and solution encoding, such as messy genetic algorithm (mGA) (Goldberg et al., 1989) and linkage evolving genetic operator (LEGO) (Smith and Fogarty, 1996). The evolutionary search process concerns thus both individual improvement and links between genes. The coupling of classical EA with a learning process makes it possible:

- to explicitly formalize links between genes and links between genes and objectives in a "model of knowledge" (MoK) distinct from the search method;
- to take into account the knowledge of experts expressed directly within the MoK.

As detailed in Section 3, the acquired knowledge can be used simply in order to give orientations to the EA by introducing some

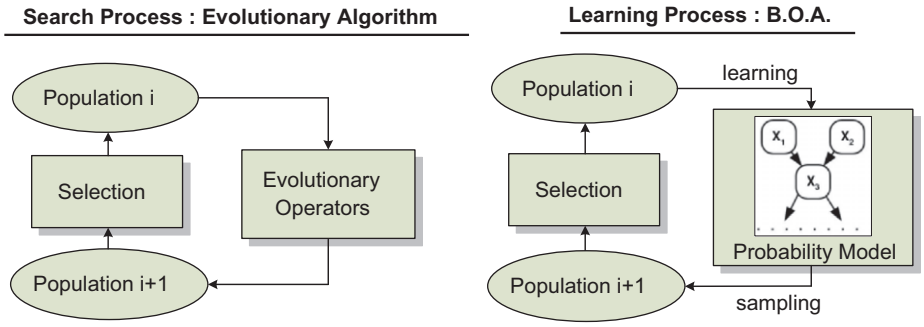


Fig. 4. Two kind of classical optimization approach.

bias into the classical evolutionary operators (initialisation, mutation and crossover operators).

2.2. Learning process

Among the different optimization methods based on a learning process, this section focuses on approaches using a probabilistic model such as population-based incremental learning (PBIL) (Baluja, 1994) and extended compact genetic algorithm (ECGA) (Harik et al., 2006). Indeed, such model can provide guidance with approximate information directly usable in order to give orientations to search operators. Bayesian optimization algorithms (BOA) (Pelikan et al., 1998) uses Bayesian networks (BN) as a model of knowledge (Fig. 4). In this method, the MoK is learned from a sample set containing selected individuals from the previous generation (according to their fitness or performance). Then, a sampling procedure is used to generate, directly from the MoK, the new population of individuals without using modification or combination operators. One of the main characteristic of the learning process lies in the construction of the sample data set. A substantial set of individuals, distributed in the promising areas is essential to obtain a relevant model. When the individuals used as a learning set are selected, the induction of the probability model, especially parameters interaction (i.e. definition of the Bayesian network structure), constitutes the hardest task to perform automatically (Baluja, 2002). Therefore, classical BOA learning process focuses on the study of most influent parameters interaction.

Another way to address this difficult problem is to use a prior knowledge in order to improve the learning procedure. The use of prior knowledge allows either to speed up algorithm convergence by introducing some high-quality or partial available solutions (Schwarz and Ocenasek, 2000), or to improve the learning procedure using an available structural knowledge (prior probabilities of networks structure) (Schwarz and Ocenasek, 2000; Baluja, 2002; Hauschild et al., 2008). The learning model proposed in this paper (Section 3) relies on the acquisition, from experts, of "a priori" knowledge about the structure of the network. Therefore, during the optimization process, the learning is achieved only through probability updates. This method makes it possible to concentrate the learning effort to the probabilities estimation.

In the proposed approach, the different objectives are considered separately in the MoK (non-aggregative approach). That makes it possible to dissociate the knowledge related to the different parts of the surface of compromise between objectives and then to propose a specific guidance towards each zone of the search space.

2.3. Integration of search and learning

In the two previous types of methods, the computing time is used either for evolutionary search process or for learning process. The choice between both approaches depends on the evaluation cost of individuals (time), the number of genes and the complexity of interactions between genes. On one hand, if the evaluation cost of an individual is very important, the traditional EA is less powerful. On the other hand, for a model containing a great number of variables or complex interactions, the knowledge learning cost may be prohibitive for the learning algorithms. The coupling of both approaches allows restricting the search process to the areas with good performances and allows the manipulation of complex configurations of genes by means of biased evolutionary operators.

In the approaches found in the literature, the two processes (search and learning) have few interactions during execution, especially for the crossover operator. Most of them, such as the learnable evolution model (LEM) (Michalski, 1998) or intelligent optimization method (Huyet and Paris, 2004), alternate between:

- a learning and sampling phase that produces "genetically engineered" individuals by instantiation of the learned knowledge model,
- a classical evolutionary phase that randomly combines and modifies the set of individuals.

Sebag and Ravise (1996) propose an original approach where the learning process also provides rules to characterize effective crossover or mutation (crossover or mutation rate, type of operators: uniform crossover, N-point crossover, etc.).

For the majority of IEO methods, the use of knowledge is achieved indirectly. This generates a black-box effect incompatible with an expert utilisation of the knowledge model. Knowledge can be represented by means of operator classes (Sebag and Schoenauer, 1994), intervals (Michalski et al., 2006), assumptions on the parameters values or by the attributes about good solutions (Chung, 1997). Huyet and Paris (2004) propose to model directly the knowledge using classes of parameters. They provide a hierarchy of parameters according to their impact on the fitness improvement. Furthermore, no model enables to dissociate objectives in order to have a representation of the influence of decisions on each of them. Objectives are aggregated (Jourdan et al., 2005) and then, partial knowledge is impossible to reuse. The proposed approach aims at delivering to the decision maker, in addition to the best solutions obtained, a MoK characterizing efficient individuals. The model proposed in the next section gives some answers to the issues listed above.

3. Proposed framework and algorithm

3.1. General architecture

The proposed framework uses a hybrid method that mixes an evolutionary algorithm for the multi-objective search process and

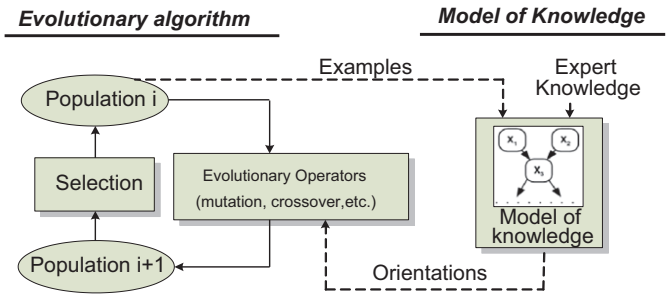


Fig. 5. Proposed global architecture.

a model of knowledge (MoK) able to provide orientations adapted to the treated case (Fig. 5). The Bayesian network (BN) formalism is used to represent the MoK. Two sources of knowledge are used to elaborate the MoK: on one hand, a selection of individuals (solutions) provided by the EA and, on the other hand, expert knowledge mainly used in order to define the structure of the BN. The resulting BN can be used by the EA as orientations for its search process. These orientations are taken into account directly by means of the evolutionary operators. Using a selection of individuals obtained during search, a learning step enables the BN to be updated by means of an inference learning algorithm.

3.2. The model of knowledge

The aim of the model of knowledge (MoK) is to formalize links between three spaces: the decision space, the evaluation space and the context space. The decision space relates to all the decisions that can be taken by decision makers, whether design or project choices (see Section 1). The evaluation space concerns the objectives to reach by the search method and, more precisely, the performance of the solutions with respect to these objectives. To be able to reuse knowledge from previous experiences (here previous project/product), the context of each experience (e.g. project/product) has to be described using supplementary parameters. All the external parameters that can influence the search process are gathered into the context space. For example, an external parameter can be the supplier capacities which influences decision related to the choice between various suppliers for a task. The modification of these external parameters is considered as an input of the model and their influence on the two other spaces has to be taken into account.

The formalism used for building the MoK is Bayesian networks (BN) (Pearl, 1988) because of their learning capacities and practical decision aiding abilities. A BN is a probabilistic model that represents a set of variables (nodes) and their conditional dependencies (edges between nodes) in a directed acyclic graph. It allows the inference of the conditional probability of each state of a node according to the state of others nodes. Conditional probabilities are gathered into a conditional probabilities table (CPT) linked to each node. An interesting characteristic of a BN is the graphical representation (e.g. see the BN of Fig. 6) very suitable for an aided decision perspective. As illustrated in Fig. 6, the MoK contains four kinds of nodes: objective, decision, concept and environment nodes. The decision nodes correspond to the XOR connectors of the project graph. The objective nodes represent the different objectives used for multi-objective optimization. The concepts nodes are used by experts to express which characteristics of the domain are important and can influence one or several objectives. Environment nodes enable to contextualize the knowledge contained into concept nodes.

Each node of the MoK is discrete, i.e. it contains only a finite set of possible states for the modelled entity. A decision node can take

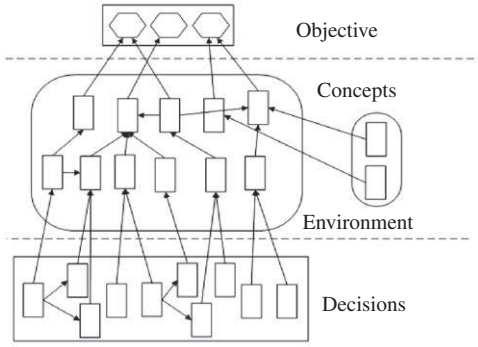


Fig. 6. Decision analysis and capitalization in the global MoK.

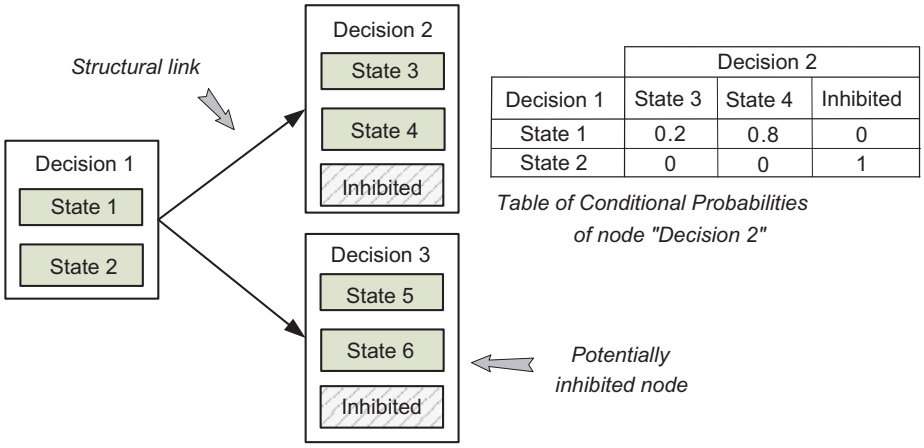
into account all the possible choices (or states) for this decision. An objective node takes into account one of the objectives to optimize (in our experiments, two objectives are considered: minimisation of cost and delay) and is represented by discrete states (e.g. *low*, *medium*, *important*). This characteristic is used in order to define different objective classes (see Section 3.2.2). States of an environment node represent the different discrete possibilities of the context that can influence the objectives. Concept nodes have two distinct roles in the MoK: (1) to reduce complexity and (2) to model expert knowledge. Under the hypothesis that the concept nodes are not necessary, it is possible to draw some arcs directly between decision nodes and objective nodes, with respect to the different influences known by the experts. The obtained model may be sufficient to capitalize expert knowledge and to guide the evolutionary algorithm. However, the complexity of such a model will be very important because it is directly proportional to the dimension of the table of conditional probabilities of the objective nodes which depends on the number of parent nodes and the number of states of those parent nodes. In order to limit the complexity, concept nodes enable to build progressively the links between decisions and objectives. In such way, the number of parents of each node is reduced, as well as the global complexity of the model. So, introducing concept nodes enables to take into account the particular influences of a limited number of decisions on sub-criteria. For instance, during a project, a great number of decisions about sub-contracting or not can be linked to one sub-criterion called "sub-contracting", represented by a concept node. This concept node can be linked to objective nodes according to expert knowledge (e.g. sub-contracting can influence the cost and the delay but not the weight of the product).

Without learning process, the probabilities of states for each decision node of the BN are uniform (i.e. the search space is considered as uniformly interesting). These probabilities can be updated by a learning procedure performed on a selection of solutions provided by the EA or by knowledge provided by the experts.

3.2.1. Structural knowledge within the MoK

The structure of the MoK is given by the nodes and the arcs between nodes. The majority of the arcs starts from decision nodes and go to objective nodes, via concept nodes. Considering the project graph (Fig. 2), some choices about design can inhibit other choices. That is taken into account in the MoK by means of arcs between decision nodes and a particular state called "inhibited" that indicates the inhibition of a decision by an upstream decision.

For instance, the simple BN represented in Fig. 7 represent three possible linked decisions (three nodes linked by two arcs). The first decision to take during this project, represented by the



	Decision 2		
Decision 1	State 3	State 4	Inhibited
State 1	0.2	0.8	0
State 2	0	0	1

Table of Conditional Probabilities of node "Decision 2"

Fig. 7. Representation of structural knowledge by means of a Bayesian network.

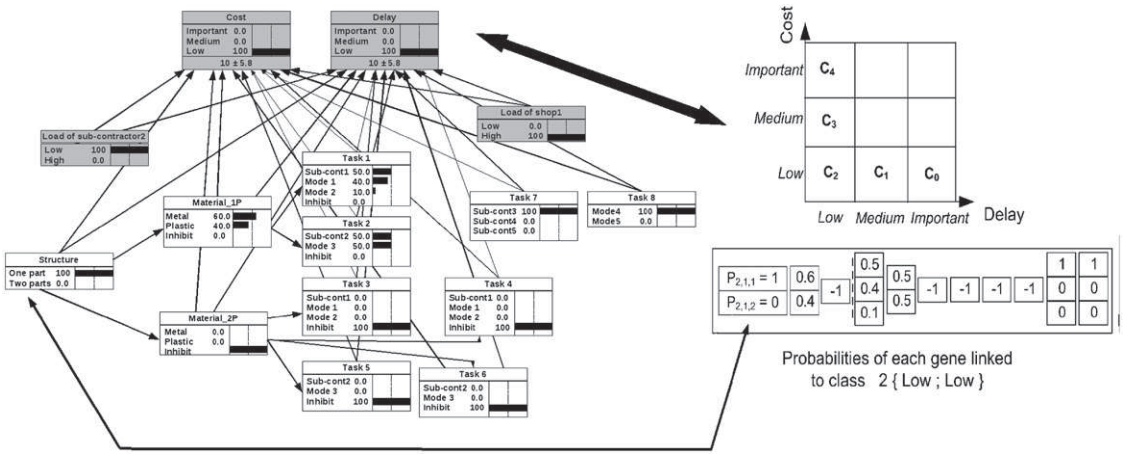


Fig. 8. Objectives classes' definition.

node "Decision 1", can be "State 1" or "State 2". If the decision "State 2" is taken, then the decision represented by the node "Decision 2" is inhibited and its state "Inhibited" has a probability of 1. On the other hand, if the decision "State 1" is taken, then the node "Decision 3" is inhibited (states "State 5" and "State 6" have a probability of 0 and the state Inhibited has a probability of 1). The tables of conditional probabilities of nodes called "Decision 2" and "Decision 3" enable to represent the different probabilities to have a particular state for a decision with respect to the states of its parent.

3.2.2. Objective classes within the MoK

In order to be able to guide the evolutionary search process, the MoK has to be representative of different objective classes. The number of classes is obtained from the number of discrete states of the objectives. In fact, only particular zones of the objective space are interesting in order to guide the evolutionary algorithm. Fig. 8 represents a MoK and the associated objective space with two objectives and three discrete states by objective (there are two environment nodes (suppliers and workshop capacities) and no concept nodes). Nine areas are defined but only five of them are interesting (C_0 to C_4). In a multi-objective optimization process, the method has to provide decision makers with a set of solutions belonging to the Pareto front. A good quality of this set is obtained when all the objective classes corresponding to the Pareto front have at least one solution. So, the proposed method enables to guide the EA to reach, at each generation, an ideal Pareto front or, more exactly, interesting zones of search space represented by the different classes of objectives (Fig. 8).

In order to use the knowledge capitalized into the MoK, it is necessary to compute probabilities linked to the different objective classes. Therefore, in the BN, fixing probabilities of some objective states to 1 enables to obtain probability of each state s of a decision d with respect to the objective class c (noted $P_{c,\ d,\ s}$). This operation consists in setting some evidences in the objective nodes (setting probabilities to 1) in order to obtain the probability of each state in each decision to reach the zone of the objective space defined by its class. For instance, let consider class C_2: probabilities to have a "low" cost and a "low" delay are considered as certain and probabilities are set to 1 (see Fig. 8).

Let $\{s_{1d}, s_{2d},\ldots,s_{kd}\}$ be the different possible states for s (s_{kd} is the inhibited state of the decision d, if it exists in the node). If $P_{c,d,Skd}=1$ then $P_{c,d,s}$ is set to -1 for each $s\in\{s_{1d}, s_{2d},\ldots,s_{kd}\}$. Of course, the value of $P_{c,\ d,\ s}$ is not interpreted as a probability in this case, but this value indicates that the decision d is inhibited for the class c.

3.3. The evolutionary algorithm oriented by knowledge (EAOK)

The search algorithm (right side of Fig. 9) is adapted from a SPEA method (strength Pareto evolutionary algorithm, illustrated on left side of Fig. 9) proposed by Zitzler and Thiele (1999). The modified SPEA proposed in this paper is based on this traditional EA with classical steps: initialisation, evaluation/archiving, selection and

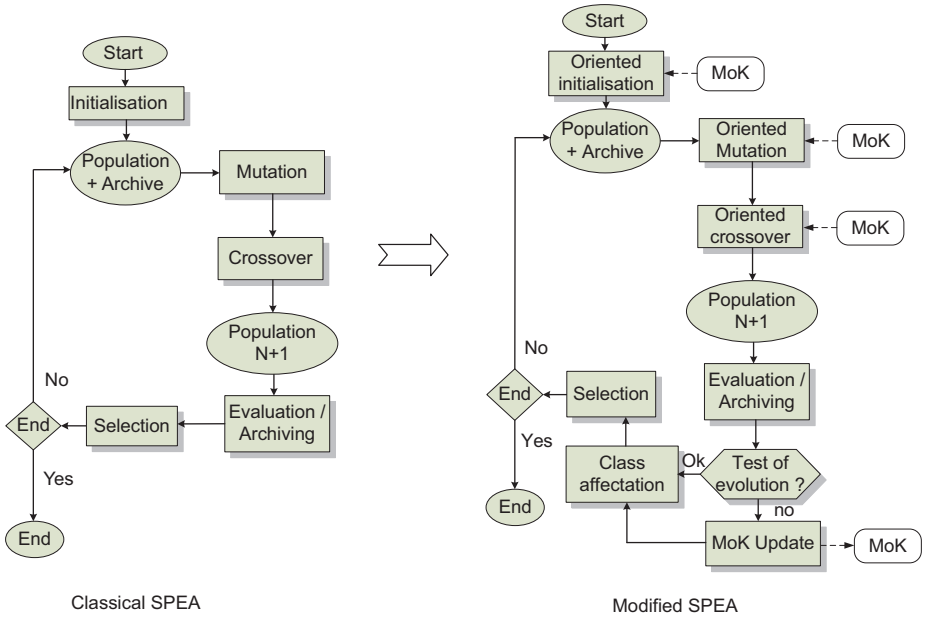


Fig. 9. Evolution of SPEA method.

genetics crossover and mutation operators. The evaluation, archiving and selection operations have not been modified.

SPEA ensures the multi-objective evaluation of individuals according to two steps: (i) the cost of solution is computed for each criterion (e.g. global cost and delay); (ii) then, the multi-criteria evaluation is achieved by means of a Pareto front to compare and classify the solutions. The probability of selection of an individual is proportional to its performance (called "fitness"). An individual's fitness depends on its position in the search space compared to the Pareto front. The fitness of an individual i is given by formula (6) (Zitzler and Thiele 1999) according to the strength (S_j) of individuals j that dominate i (an individual j that dominates i is noted $j > i$, and correspond to the fact that for each criterion, the performance of the individual j is equivalent or better than the performance of the individual i). The strength of an individual S_i is given by formula (7) where n is the number of dominated solutions and $|Pop|$ is the population size.

$$f_i = \frac{1}{1+\sum_{j,j>i} S_j}. \tag{6}$$

$$s_i = \frac{n}{|Pop|+1}. \tag{7}$$

3.3.1. Solution encoding

In order to define the new EA, an encoding of a solution (a scenario) is needed. The model proposed by Baron et al. (2004) is well suited for the representation of a project scenario. Therefore, it is used in the proposed EA. A project graph and an individual corresponding to one scenario are represented in Fig. 10.

The chromosome of an individual gathers on the left side the genes corresponding to decisions derived from product decomposition (choices between components represented by XOR nodes in the project graph). Instantiations of the genes of this first part (selection of a particular state) can lead to the inhibition of some other genes in the chromosome. On the right side of the chromosome, genes represent decisions derived from project decomposition (choices to achieve tasks on the graph, represented by dotted circles on Fig. 10). A gene g represents a decision d. A value of a gene g represents a choice (state) s for a decision d. All the possible choices are always represented even if several of them are inactive since they are inhibited by choices made on genes of the left side of the chromosome. This encoding ensures a constant viability of the solutions. This aspect of inhibited variables has been previously studied by Paris et al. (2001). But in this study, the authors traduce it by a specific coding (individuals represented by trees) and specific evolutionary operators that allow handling this tree representation.

During the execution of the EAOK, two strategies can be used: "structural knowledge utilisation" and "diploid knowledge preservation". These two strategies are presented below.

3.3.2. Structural knowledge utilisation

During the execution of the EAOK, if a gene is inhibited by a previous gene instantiation, the values of the corresponding probabilities associated to the objective class ($P_{c,\,d,\,s}$) are set to -1 indicating its inhibition. In Fig. 11, the fact that $P_{2,1,1}=1$ leads to the inhibition of third gene, then the inhibition of this gene also leads to the inhibition of others depending genes (here, genes 6–9). This inhibition mechanism corresponds to the mode called "structural knowledge utilisation". It can be applied all along the algorithm or not applied, according to the strategy chosen for the algorithm (see Sections 3.3.5 and 3.3.6).

3.3.3. Diploid knowledge preservation

During the execution of the EAOK, different operators can modify the genes of the chromosomes. A mode called "diploid knowledge preservation" permits to preserve inactive genes which can represent interesting characteristics in other areas of the search space and that can be re-activated during other cycles of the EAOK. This biological concept was already used successfully for optimization in dynamic environment by Holland (1975). During optimization process, if "diploid knowledge preservation" mode is activated, inactive genes are not modified (see Sections 3.3.5 and 3.3.6).

3.3.4. Initialisation oriented by knowledge

The first step of the EAOK is the initialisation of individuals of the population according to the probabilities $P_{c,\,d,\,s}$ of decisions with respect to each objective class. The initial population is built

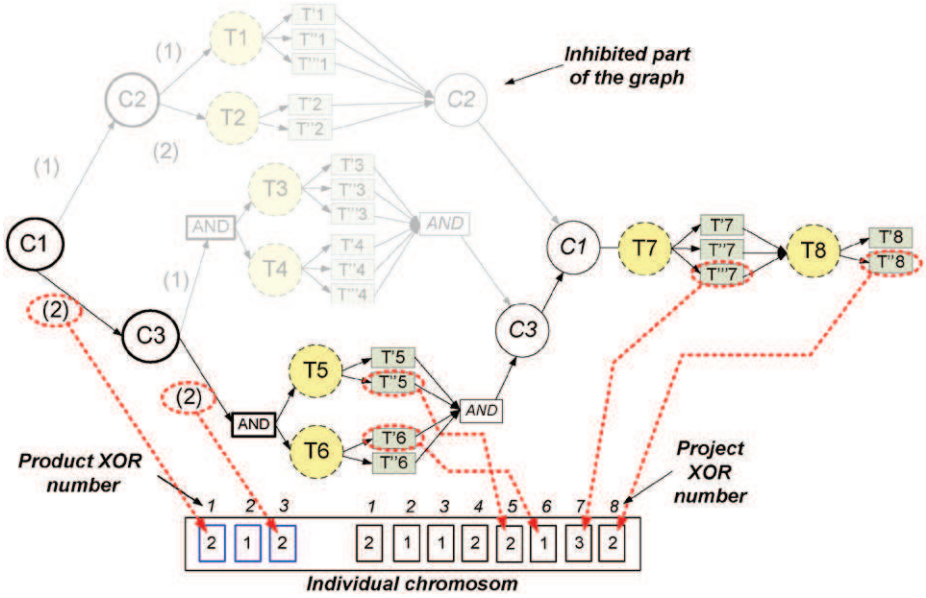


Fig. 10. Scenario encoding.

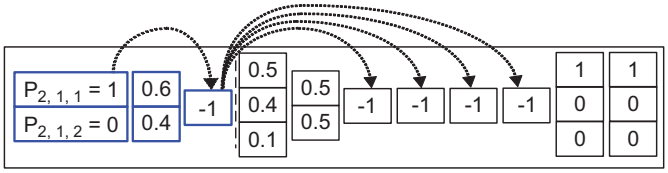


Fig. 11. Example of structural knowledge utilisation.

according to the objective classes in order to start the search procedure with a priori good orientations. The initial population (the constant size N of the population is a parameter of the EAOK) is created with individuals uniformly distributed to the different objective classes (individuals are randomly assigned to objective classes). After assignation to objective classes, the values of the individual genes are fixed using the probabilities of individual classes in order to give a priori good orientations. As illustrated in Fig. 12, the probability to select a value g_0 (model of state s) of a gene g (model of decision d) belonging to an individual i associated to a class c is given by: $P_{c,\ d,\ s}$. Therefore, values of genes are selected by a roulette wheel selection (RWS) mechanism. If $P_{c,\ d,\ s}$ is equal to -1, then the choice of the gene value is not important and a random selection is done.

3.3.5. Mutation oriented by knowledge

First, the mutation operator selects an individual randomly among the population according to the probability of mutation P_{mut} (input parameter of the EAOK). Then, as illustrated by Fig. 13, the probabilities of the individuals' objective class are used to fix the value of genes. For a gene g, two cases are taken into account:

- g is inactive: g is not muted if "diploid knowledge preservation" mode is chosen. If this mode is not chosen, g is selected for mutation (or not) according to P_{mut} and, if selected, g is muted and its value is chosen according to a random selection between possible states;
- the gene is active: g is selected for mutation (or not) according to P_{mut} and, if selected, g is muted and its value is selected using RWS according to the probabilities of its objective class.

The last step concerns structural knowledge utilisation. If some genes are inhibited by mutation of previous genes, the associated values of the objective class are set to -1.

3.3.6. Crossover oriented by knowledge

The operator selects a first individual (first parent noted i_1) randomly among the population according to the probability of crossover P_{cross} (parameter of the EAOK). The second individual (i_2) is chosen according to the crossover strategy: exploratory or intensification strategy. The exploratory strategy consists in choosing the second parent associated to another objective class (firstly in the classes closest to the class of individual i_1). The intensification strategy consists in choosing a second parent associated to the same objective class. Once parent selection is done, probabilities of their classes are used to determine the points of crossover (an example of crossover operation is illustrated on Fig. 14). The crossover is performed in a specific manner for each individual (unilateral crossover).

Considering a value g_0 (state s) of a gene g (decision d) belonging to the selected parent i_1 associated to the class c, the crossover is performed according to the probability $(1-P_{c,d,s})$ (also called pertinence of the active state) if $P_{c,d,s} \neq -1$. The crossover consists in copying the corresponding value of the gene of parent i_2 into the current gene of the parent i_1.

This method makes it possible to preserve and, if possible to exchange, favourable genes of each individual. When the value linked to a gene in the corresponding objective class is -1 (inhibited gene), a unilateral crossover is done with a probability of 0.5 if the "diploid knowledge preservation" mode is inactive (uniform crossover of inhibited genes). If the "diploid knowledge preservation" mode is active, inhibited genes are preserved from the evolutionary process.

3.3.7. Test of evolution

During EAOK execution, the MoK is not updated as long as the best solutions of the Pareto front are improved at each cycle. If the search process does not evolve, two reasons are possible: (1) the global optimum is reached or (2) a local minimum has been found. For the second possibility, the main cause can be erroneous, unsuitable or incomplete probabilities in the MoK. Therefore, if there is no evolution after a predefined number of cycles of the

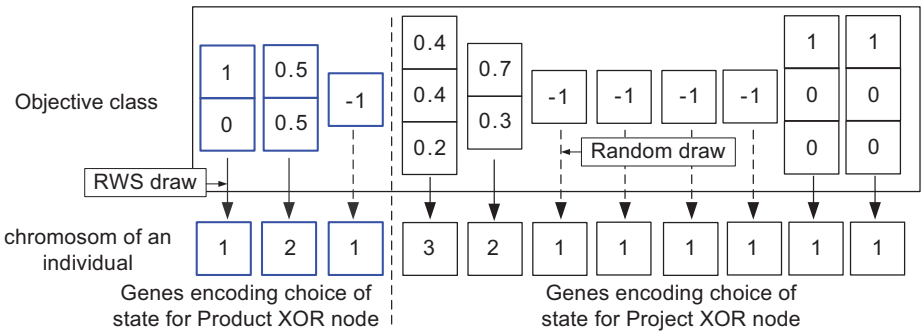


Fig. 12. Example of initialisation oriented by knowledge operator.

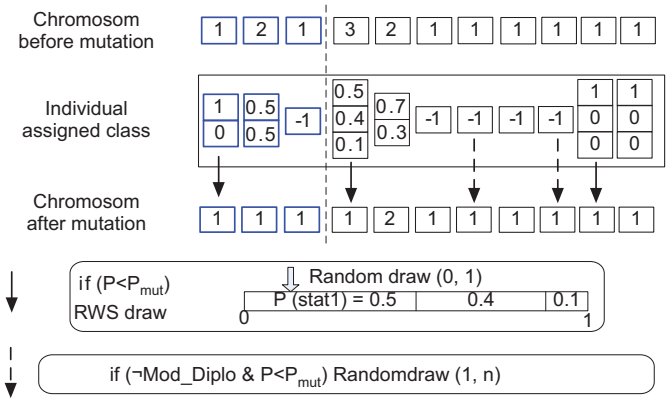


Fig. 13. Example of mutation oriented by knowledge operator.

EAOK, then the MoK is **updated** by means of: (1) a probability smoothing procedure, (2) a learning procedure from a selection of individuals. Both permit to make evolving the MoK for a better orientation of the EAOK and are described in the next sections.

3.3.8. *Probability smoothing*

In order to modify the MoK, the probability smoothing process is used at first. It consists to change the probabilities of the BN according to the formula (8) below

$$P' = P \times (1 - smooth\_degree) \tag{8}$$

where P is a probability before smoothing and *smooth_degree* is a parameter that permits to control the smoothing process. P' is the probability after smoothing. The smoothing degree is included into the interval [0, 1]. If *smooth_degree* is equal to 1 the operators are not oriented by knowledge (EAOK degenerates as a classical EA). If *smooth_degree* is equal to 0, there is no probability smoothing.

3.3.9. *Learning procedure*

Probabilities of the BN are learned from representative cases using EM algorithm (expectation–maximization) (Dempster et al., 1977 Tanner, 1996). The EM algorithm can be used in BN for finding maximum likelihood estimates of parameters. The model can depend on unobserved latent variables. So, it is interesting for the learning process because cases can be incomplete (notably for a partial knowledge reuse). A case represents an individual with the values of concepts, criteria, decisions and associated objectives. EM algorithm stops the learning process following two criteria: the number of iterations E–M or the improvement of the BN likelihood compared to the previous learning cycle (network quality indicator with respect to the set of learning examples).

3.3.10. *Affectation of the individual to objective classes*

Individuals created by means of crossover and mutation operators have to be affected to objective classes in order to start a new cycle. An individual is affected to its closest objective class. In the objective space, for each objective class, a central individual is defined (Fig. 15). Its role is to represent the objective class tending to attract new individuals. Central individuals belong to the current Pareto front.

3.4. *Computational complexity of EAOK*

The proposed algorithm is a modified SPEA complemented by a BN guidance (that involve learning and inference algorithm). The computational complexity of the traditional SPEA method is $O(KN^3)$ (Zitzler and Thiele, 1999), where K is the number of objectives and N the size of population. The modified genetics operators added to the classical SPEA method (evaluation and *KO*-operators) have a bounded complexity proportional to the number of decision variables of the problem. The BN guidance, beforehand computed ("a priori" orientations) then updated after every learning phase or probabilities smoothing phase, is obtained by a learning algorithm (EM algorithm) on the set of selected individuals. The learning phase has a polynomial time complexity which depends on the number of learning cases (individuals), the selected stopping criterion and the BN shape (especially the number of un-instantiated nodes—decision and concept nodes), the number of states of those nodes and their relationships (number and size of clique in the BN) (Dempster et al., 1977). Finally, the inference algorithm used to exploit the learned model (classes acquisition) is the junction tree algorithm. It is done for each objective class on the learned model. Its time complexity also depends on network shape (see Cowell et al., 1999) for more details on inference and learning algorithm). So the complexity of the whole algorithm is thus polynomial.

4. Experimentation and results

The main contribution of this study consists in the hybridation of an evolutionary algorithm with a dedicated model of knowledge. This knowledge takes three distinct forms: (1) a conceptual dependency structure between parameters expressed by a Bayesian network, (2) probabilities extracted from analysis of previous optimizations and, (3) explicit structural knowledge (inhibitions between genes). To evaluate the use of each type of knowledge, the behaviour of three algorithms is studied in this section:

– classical EA (without knowledge): EA is ran with equiprobable objective classes, in such way that each state has an equivalent probability to be mutated or crossed. The main specific features in this case are the use of the inhibition

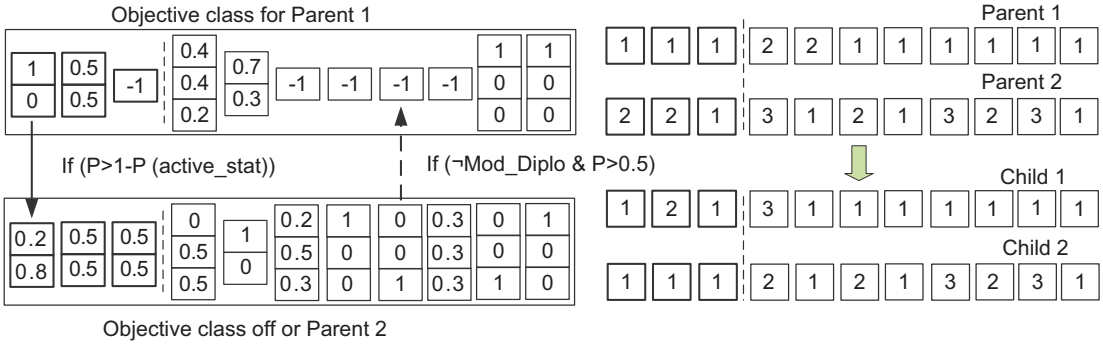


Fig. 14. Example of crossover oriented by knowledge operator.

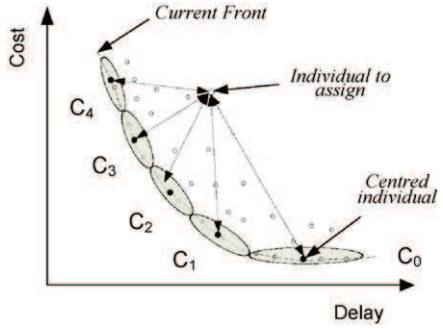


Fig. 15. Individual affectation to objective classes.

mechanism and of the crossover strategies (intensification and exploration),

- knowledge oriented algorithm using «on line» learning (noted $EAOK_X$): the project graph structure is defined at the beginning of the optimization while probabilities tables, initially uniforms, are progressively updated by learning every X generations,
- knowledge oriented algorithm guided by a pre-learned model (noted $EAOK_{init}$): the BN structure given by experts is used while probabilities are learned using a sample of best solutions found previously with an exhaustive approach (for small instances) or with previous executions of $EAOK_X$ (for larger instances).

The $EAOK_{init}$ algorithm enables to check the impact of using all the available knowledge (structure and probabilities) while the $EAOK_X$ only uses the minimal necessary knowledge (BN structure given by an expert). The analysis of the three algorithms should allow choosing a control strategy according to the available knowledge and its relevance.

The algorithms have been studied by means of an ad hoc platform developed in C++. Experimentation has been planned in two steps. In the first step, the algorithms are evaluated on limited size problems (different graph shapes with 35 to 90 task nodes and 10 to 40 XOR nodes using a test graph random generator, see next section). This first step allows checking the general behaviour of the algorithm as well as tuning of several parameters (evolutionary parameters, crossover strategies, learning parameters,…). In a second phase, the behaviour of the algorithms is studied on a larger project (approximately hundred XOR nodes and three hundred task nodes in the project graph).

4.1. Test graph random generator

The main principles of the graph generator used in order to build the test graphs are described in this section.

The graph shape is randomly generated by a recursive algorithm which treats graph by subsets in which the objects to assign (tasks, XOR, AND nodes) are distributed. It allows obtaining a nearly balanced graph with a similar number of nodes in every subset. We also use, for the first test phase, some particular graph shapes (linear or tree shape).

The graph generator has been achieved with the intention of introducing underlying knowledge into data. This allows to simulate a coherent knowledge representation that can be spotted by the learning algorithm. Therefore, some "concepts" are attached to each XOR node, which modulate the performance of associated tasks (tasks located on the branches of the XOR node) for each criterion.

4.2. Global evaluation of the strategies

Fig. 16 and Tables 1–5 introduce first tests results on different small projects (35 task nodes randomly generated, 12 XOR nodes for Fig. 16 and Table 1 for example). The curves at the top of Fig. 16 show the average performance of the population of individuals obtained with the strategies EA, $EAOK_{init}$, $EAOK_1$ and $EAOK_5$. The curves at the bottom of Fig. 16 show the average performance of the individuals of the Pareto front. Each curve represents average values obtained after one hundred executions.

The performance of a scenario is equivalent to the global fitness F (noted *fitness* below). F corresponds to the sum of normalised objective values. For each criterion, the minimum and the maximum experimental values permit to obtain a normalised objective value in order to be added to the other ones.

$EAOK_{init}$ shows good performances. After initialisation, individuals of the population are 25% better than those obtained with EA. These results come from different combination of others parameters detailed in Sections 4.2–4.5 (crossover strategies, knowledge use, etc.). This explains the important standard deviation but, for a given setting, the performance ratio between EA and $EAOK_{init}$ is stable. The initial gap between EA and $EAOK_{init}$ corresponds to the direct impact of knowledge injection at the initialisation step. This gap varies according to the MoK quality. The population generated by the guided EAOK is always improved in comparison with classical EA, because the used MoK leads to a concentration of the population within good performance areas. The final Pareto-optimal individuals mean fitness is improved of 4.82% at the twentieth generation. EA performance meets $EAOK_{init}$ ones very progressively, according to the problem complexity (number of parameters and complexity of injected knowledge) and according to evolutionary parameters setting.

Fig. 16 presents the first tests for on line learning algorithms. They are equivalent to EA at the beginning of optimization process (uniform probabilities distribution). They deviate from EA after every learning phase. He learning effect is particularly visible in mode $EAOK_5$ with three zones where the difference with EA is

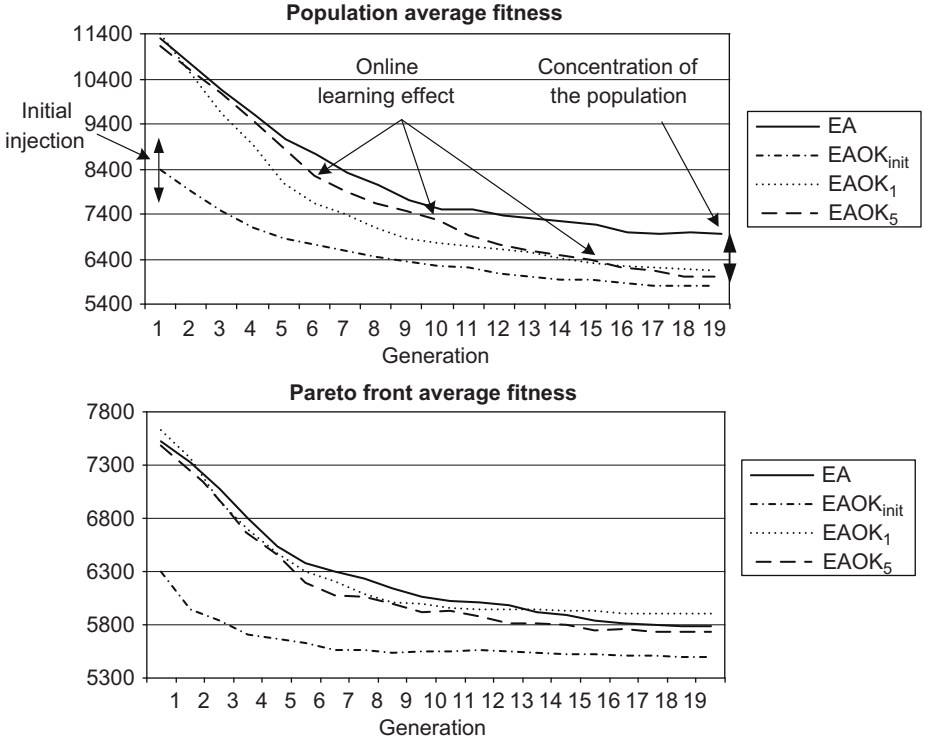


Fig. 16. Population and Pareto front average fitness. Test realized for 20 generations, with a population of thirty individuals, a maximum of nine individual in Pareto front, P_{mut}=0.5, P_{cross}=0.5 and five classes of objectives.

Table 1
Values corresponding to the curves showed on Fig. 16. The table shows results obtained for each strategy (value and relative standard deviation (RSD) in percentage for one hundred executions).

Mode	Mean fitness for entire population				Mean fitness for Pareto individuals			
	Generation 0		Generation 19		Generation 0		Generation 19	
	Value	RSD	Value	RSD	Value	RSD	Value	RSD
EA	11289	4.4	6961	16.5	7521	13.9	5781	7.5
$EAOK_{init}$	8408	3.9	5815	11.8	6305	10.3	5502	4.9
$EAOK_1$	11352	4.4	6130	27	7624	13.7	5900	10.9
$EAOK_5$	11144	4.4	6010	30	7502	14.5	5728	10.6

increased (generations 5, 10 and 15). At the beginning of the process, $EAOK_1$ has better performance than $EAOK_5$ but, the difference is progressively reduced and finally $EAOK_5$ gives better results. This can be explained as follows: $EAOK_5$ leaves degrees of freedom to the search process in order to improve individuals between each learning phase, whereas, for $EAOK_1$, individuals selected for learning are not enough diversified and the guided search remains in a restricted area of the search space. $EAOK_5$ better takes advantage of search and guiding combined effects.

4.3. Crossover strategies evaluation

The average values for one hundred executions of EA and $EAOK_{init}$ with two crossover strategies (exploration and intensification strategies respectively noted C_{explo} and C_{int}) on the same project are represented in Table 2.

During optimization, the performances of EA, whatever the selected crossover strategy, are similar. With the exploration strategy, individuals have a good distribution on the Pareto front. However, they give individually worse performances. On the contrary, when using the intensification strategy the individuals are gathered and their performances are better. Let us point out that at the end of the optimization process the strategy of intensification is globally more powerful than exploration (average values of every mode). However, the EA with an intensification strategy alone gives worse performance than the EA with an exploration strategy only. This can be explained by the lack of diversity of the population using a strategy of intensification during all the process. With respect to the relative standard deviation (RSD), the intensification strategy always achieves better results with, for example for the best individual of $EAOK_{init}$, a RSD value of 0.6%.

The most visible effect of the choice of crossover strategy affects the average improvement given by the oriented crossover operator. $EAOK_{init}$ with a strategy of intensification presents a significant initial peak, corresponding to the fast improvement in all the known good performances areas. The average performance of the crossing for the first generation is 233 for $EAOK_{init}$, while EA improvement reaches only 29.5 in exploration mode. The individuals are thus correctly crossed, by respecting the integrity of relevant knowledge of each class, and then produce individuals mixing good features of both parents. For the whole process, the crossover improves of 41 the global fitness of the solutions for the strategy of exploration whereas the improvement in intensification strategy is −18.2. During the following tests, an exploratory strategy has been used during the optimization process, notably in order to provide a set of diversified solutions to the learning process in mode $EAOK_X$. The intensification strategy is used only at the end of the search process in order to refine obtained solutions.

4.4. Learning parameters setting

For the tuning of the learning algorithm parameters, two important characteristics must be taken into account: the quality of the examples used for learning and the stopping criterion of the

Table 2
Crossover strategies evaluation (values and RSD in percentage) for EA and $EAOK_{init}$ modes, and for the first and last generation on the same small project.

Mode		Mean fitness for entire population				Mean fitness for Pareto front individuals			
		Generation 0		Generation 19		Generation 0		Generation19	
		Value	RSD	Value	RSD	Value	RSD	Value	RSD
EA	C_{int}	11289.2	4.4	6960.8	16.4	7520.7	14.0	5781.4	7.5
	C_{explo}	11369.38	4.2	7108.7	13.0	7562.1	12.6	5682.6	7.5
$EAOK_{init}$	C_{int}	8407.609	3.9	5814.5	11.8	6303.9	10.3	5502.6	4.9
	C_{explo}	8401.002	3.4	5845.0	8.5	6307.5	10.4	5545.8	5.3
Mode		Mean fitness for current best individual				Average improvement with crossover operator			
		Generation 0		Generation 19		Generation 1		entire process	
		Value	RSD	Value	RSD	Value	RSD	Value	RSD
EA	C_{int}	6193.2	15.3	4064.2	3.4	10.2	1791	−40	110
	C_{explo}	6255.3	14.8	4087.8	3.7	29.5	662	−32	149
$EAOK_{init}$	C_{int}	4237.2	5.1	3985.7	0.6	4.5	2521	−18.2	138
	C_{explo}	4240.8	5.5	3996.4	0.9	233	95	41	162

Table 3
Learning parameters tuning: the table presents average results for different example selection strategies: average fitness and RSD (in percentage for one hundred runs) of individuals of the population, Pareto front at the end of the optimization, average fitness of the best obtained individual and finally average execution time.

Mode		Mean fitness for entire population		Mean fitness for Pareto front		Mean fitness for best individual		time
		Val.	RSD	Val.	RSD	Val.	RSD	
EA		6453	8.4	5177	3.6	4272	2.2	14
$EAOK_{10}$	Sel_1	6142	8.6	5213	3.5	4282	2.7	37
	Sel_2	5633	7.5	5178	2.3	4268	1.7	17
	Sel_3	5556	6.9	5157	2.2	4255	0.5	14.5
	Sel_4	5972	9.5	5222	4.3	4288	3.1	24
	Sel_f	5517	6.6	5149	2.2	4254	0.2	17

Table 4
Learning parameters tuning: the table present average results for different stopping criterion and progressive smoothing of MoK: average fitness and RSD (in percentage for one hundred runs) of individuals of the population, Pareto front at the end of the optimization, average fitness of best obtained individual and average execution time.

Mode		Mean fitness for entire population		Mean fitness for Pareto front		Mean fitness for best individual		time
		Val.	RSD	Val.	RSD	Val.	RSD	
$EAOK_{10}$	10 iterations	6115	8.8	5211	5	4268	1.8	15.8
	0.1% log-L	6025	8.6	5174	3.3	4267	1.4	15.2
	0.01% log-L	6126	8.3	5221	3.3	4295	3.5	15.6
	0.001% log-L	6194	9.2	5218	3.4	4287	2.9	15.2
	No Smoothing	6104	8.6	5229	4	4283	2.8	15.2
	Smoothing	6127	7.1	5217	2.9	4276	1.6	15.7

learning algorithm. Both parameters are studied in the following sections.

4.4.1. Influence of the set of examples used for learning

Table 3 presents results obtained for the five following strategies on a small size project (50 task nodes):

- sel_1: this strategy selects all the individuals generated since the beginning of optimization, with removal of the duplicate individuals,
- sel_2: this strategy uses sets of individuals selected according to their performance (one set for each objective, containing the individuals which performance for the selected objective ranges between the best performance obtained so far for this objective (*bst*) and up to twenty five percent in addition to this value: *bst*+25%.*bst*),
- sel_3: this strategy uses sets of individuals of fixed size (the best individuals for each objective since the beginning of optimization),
- sel_4: this strategy uses the entire population of last generation,
- sel_f: this strategy is obtained from the strategy sel_3 and complemented with the best individuals found for the compromise between objectives.

The quality of the available learning cases set seems to be the most important characteristic in order to obtain a relevant model. The first observation is that strategies sel_1 and sel_4 lead to poor performances. Indeed, they select individuals from the entire search space including individuals of poor performance. The others strategies focus on good individuals which permits to avoid a saturation of the learning capabilities of the MoK with poor individuals.

Comparing strategies sel_2 and sel_3, the strategy sel_3 leads to better results: (i) it better selects individuals whatever the shape of the Pareto front (concave or convex front); (ii) it allows having a constant number of learning examples, which makes it possible to control precisely the processing time of the EM algorithm by defining the size of the sets of selected individual. This strategy (sel_3) was finally complemented by additional set (sel_f) in order to

Table 5
Average fitness of individuals of the Pareto front at the beginning (generation 0 to 2), in progress (generation 10 to 12) and at the end of the optimization, with various indicators allowing to evaluate more precisely the Pareto front quality: relative standard deviation (RSD) of the average fitness of individuals of the Pareto front (PD), RSD of distance between two consecutive individuals (DI), overall length of the Pareto front (Lg) and number of individuals of the Pareto front (Nb). The last column presents the average fitness of the best final individual.

Mode	DKP/KS	Average fitness of Pareto front individual								PD	DI	Lg	Nb	Best
		0	1	2	10	11	12	18	19					
$EAOK_{init}$	1/0	1667	1465	1445	1359	1348	1344	1360	1369	0.08	0.20	34	8.6	656
	1/1	1706	1545	1465	1384	1390	1400	1399	1396	0.07	0.29	32	8	659
	0/1	1665	1527	1492	1356	1357	1355	1379	1383	0.08	0.24	33	8.3	657
	0/0	1716	1484	1409	1326	1321	1336	1353	1355	0.08	0.20	34	8.6	658
EA	−/0	2322	2215	2032	1541	1522	1518	1451	1439	0.1	0.26	31	6.9	665
	1/1	2309	2148	1925	1537	1528	1510	1464	1464	0.11	0.24	30	6.9	674
	0/1	2377	2083	1984	1667	1627	1584	1461	1459	0.1	0.25	31	6.7	674
$EAOK_{10}$	1/0	2505	2255	2032	1570	1498	1483	1369	1363	0.07	0.26	32	7.4	669
	1/1	2324	2104	2008	1468	1421	1419	1364	1357	0.08	0.26	31	7.1	661
	0/1	2270	2091	1889	1562	1539	1475	1406	1399	0.1	0.25	31	7.5	669
	0/0	2380	2183	2095	1606	1545	1500	1398	1391	0.07	0.26	32	7.4	664

better learn the zones of compromise between objectives. This last strategy leads to the best results and has been selected for the remaining experimentations.

4.4.2. Influence of stopping criterion

Table 4 presents tests realized on the same graph than in section 4.3.1 (50 task nodes) for the other parameter of learning algorithm: the stopping criterion. As previously presented, two kinds of criteria are experimented: a fixed number of EM iterations (ten iterations has been tested) or the minimal improvement of log-likelihood of BN compared to the previous EM step (three different values investigated: a minimal improvement of 0.1%, 0.01%, and 0.001% to continue the learning process). The first one allows to control the computing time needed for the learning algorithm but the quality of probabilities estimation is not guaranteed. The values presented in Table 4 have been obtained with the final individual selection (sel_f) and exploration crossover strategy (except the two last lines of the table, see details below). As shown in this table, the best strategy seems to be the less restrictive one (0.1% of minimal log-likelihood improvement). Indeed, a fast learning is sufficient to make emerging the main properties of the search space and thus to obtain a global guidance, whereas a longer learning brings to an over-guidance of the search towards the existing individuals. The same conclusion can be drawn when the number of examples per class of objectives is too restricted (presented results obtained with various individual selection strategies). The phenomenon of "over-learning" induces stagnation of search around the already found individuals, with a risk of stagnation in local minima. This interpretation has been confirmed by the use of a progressive smoothing (see Section 3.3.8) of the MoK, presented by the two last lines of Table 4. The progressive smoothing can be used and provides two functions: (i) it makes it possible to limit "over-learning" when cases provided to the learning are too similar; (ii) it constitutes a mean for gradually giving degrees of freedom to the search process, i.e., to limit the guidance by the MoK (phenomenon of "over-guidance"). If smoothing gives good results on reduced size projects, it nevertheless requires more computing time, in particular for bigger ones. Thus, it should not be systematically used but only as a last resort when the optimization stagnates.

4.5. Structural knowledge and diploid preservation mode setting

Every combination[1] of structural knowledge (SK) and diploid knowledge preservation (DKP) modes has been evaluated. The results are presented in Table 5 and concern one hundred executions of each strategy: $EAOK_{init}$, EA and $EAOK_{10}$ on a project of 50 task nodes. Structural knowledge can be used to indirectly manage the knowledge contained in the individuals. If it allows an initial improvement of the AE, it also involves a reduction of the genetic diversity by reducing exchanges between the individuals. On the other hand, the use of structural knowledge with a MoK learned online allows using only individual specific information among knowledge contained in class. The diploid knowledge preservation mode gives good results only when the individuals have already a good level of performance, by preserving the inactive combinations which can be re-used when the corresponding genes are re-activated. On the contrary, the best strategy with reliable information ($EAOK_{init}$) is to use neither structural knowledge, nor diploid knowledge preservation. Guidance by the model is then complete, but this strategy must not to be maintained because of stagnation risks (strict guiding towards existing individuals).

4.6. Large size problem experimentation

Finally, the proposed method has been tested on a problem of larger size (350 tasks nodes and more than 100 XOR nodes). An exact algorithm is not suitable for such large project. Individuals used for the construction of the "a priori" model ($EAOK_{init}$) are collected during one execution of the $EAOK_{10}$ (390 individuals). Table 6 presents the average of 20 execution of our algorithm (30 generations of 50 individuals, $P_{mut}=P_{cross}=0.5$).

The $EAOK_{10}$ algorithm shows an interesting behaviour. The population is overall improved by guiding as well as individuals of the Pareto front. At the last generation, the variation between EA and $EAOK_{10}$, respectively, reaches 54% (population), 15% (Pareto front) and 11% (better individual) in favour of the $EAOK_{10}$. Moreover, these performances are more regular than those of the traditional EA. The learning improves the results, especially the precision and reliability of optimization. It also seems that the performances obtained strongly depend on the quality of search before the first learning. An interesting prospect is to use an adjustment of the EA supporting the diversity of individuals, in order to improve the quality of individuals provided to the learning algorithm. However, in its version of the platform, the time of inference needed to update the probabilities classes remains important. The $EAOK_{10}$ requires indeed approximately 300 seconds to reach the thirtieth generation with two learning phase, so approximately 27% of additional time required compared to the EA.

[1] Note that in EA mode, DKP is completely linked to structural knowledge activation, while in other modes, genes can be inactivated by learned knowledge.

Table 6
The table below presents average values and associated RSD (for the twenty executions) for the performance of population individuals (Pop.), Pareto front individuals (Pareto) and best individual (best) at the end of optimization process, as well as qualitative indicators for Pareto front (PD, DI, Lg and Nb) and the execution time in second.

Mode	Pop.		Pareto		PD	DI	Lg	Nb	Best		Time
	Val.	RSD	Val.	RSD					Val.	RSD	
EA	11357	0.199	6557	0.24	0.09	0.18	11.2	3.95	5453	0.21	217
$EAOK_{10}$	7348	0.20	5688	0.14	0.07	0.16	9.4	4.1	4876	0.12	298
$EAOK_{init}$	5420	0.18	3601	0.17	0.11	0.21	9.5	3.6	2953	0.11	201

5. Conclusion

This paper is focused on the description and evaluation of a new evolutionary algorithm for the selection of project scenarios in the early phases of a system design. The underlying problem is highly combinatorial especially when decisions on the product and on the project are integrated in a single model, called project graph. In order to benefit from expert knowledge and from past optimizations, a hybridation between a learning algorithm and a search algorithm is proposed. A model of knowledge, used to capitalize the knowledge that links decisions, environment, objectives and concepts, is defined using the Bayesian network formalism. This model is obtained from experts and from a learning process using some solutions generated by the EA. This model is used in order to give orientations to the EA to reach a priori interesting zones of the multi-objective space. In a decision aided perspective, the guided search process has to give some solutions well distributed on the Pareto front. The proposed method is based on the hybridation of a classical strength Pareto evolutionary algorithm in order to guide the search process by means of the model of knowledge. New operators of initialisation, crossover and mutation are defined. Their behaviour is oriented by probabilities contained into the MoK. Since the MoK can be incomplete or erroneous, a MoK updating process based on in-line learning permits to make it evolve during optimization.

Obtained results show the interest of the different levels of knowledge reuse for orientation of an evolutionary algorithm. When the knowledge contained in the model of knowledge is reliable, the proposed method allows a significant improvement of performance. When the MoK is erroneous or incomplete, the tests realised on learning algorithm enabled us to study the learning process abilities with the suggested method. To validate our approach completely, it still remains to confront it with standard problems (“benchmarks”).

However, tests carried out show the high performances of the evolutionary algorithm oriented by knowledge compared to a traditional EA. Moreover, the advantages of the proposed model relate not only to a well guided and more efficient optimization than with classical EA, but also to the possibility to capitalize knowledge about previously planned projects according to their context, and to provide decision makers with the MoK used during optimization in addition to the optimized solutions. It is indeed useful for decision makers to use the Bayesian network, thanks to the tools offered by this formalism, and to directly evaluate the influence of his decision on the objectives.

References

Baluja, S.: “Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning”, Technical Report CMU CS 95_163, Carnegie Mellon University, (1994).

Baluja, S., 2002. Using a priori knowledge to create probabilistic models for optimization. International Journal of Approximate Reasoning 31 (3), 193–220.

Baron, C., Rochet, S., Esteve D.: GESOS: a multi-objective genetic tool for project management considering technical and non-technical constraints. In: Proceedings of the IFIP World Computer Congress on Artificial Intelligence Applications and Innovations (AIAI), 2004.

Chelouah, R., Baron, C., Zholghadri, M., Gutierrez, C., 2009. Meta-heuristics for System Design Engineering, in Studies in Computational Intelligence. Ed., vol. 203. Springer, Berlin, p. 387–423.

Chung, C.J.: Knowledge based approaches to self adaptation in cultural algorithms, PhD thesis, Wayne State University, Detroit, USA, (1997).

Cowell, R.G., Dawid, A.P., Lauritzen, S.L., Spiegelhalter, D.J., 1999. Probabilistic networks and expert systems, Information science and statistics. Springer ed.

Dempster, A., Laird, N., Rubin, D., 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B 39 (1), 1–38.

Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: motivation, analysis, and first results, in Complex Systems, 3, 493-530, 1989.

Harik, G.R., Lobo F.G., Sastry, K.: Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA), scalable optimization via probabilistic modeling, 39–61, 2006.

Hauschild, M.W., Pelikan, M., Sastry, K., Goldberg, D.E.: Using previous models to bias structural learning in the hierarchical BOA, proc. of the 10th annual conference on Genetic and evolutionary computation, p. 415–422, 2008.

Holland, J.H., 1975. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor.

Huyet, A.-L., Paris, J.-L., 2004. Synergy between Evolutionary Optimization and Induction Graphs Learning for Simulated Manufacturing Systems. International Journal of Production Research 42 (20), 4295–4313.

Jourdan, L., Corne, D.W., Savic, D., Walters, G.A., 2005. LEMMO: Hybridising rule induction and NSGA II for multi-objective water systems design. Proceedings of Computing and Control in the Water Industry Conference 2, 45–50.

Li, B., Chen, L., Huang, Z., Zhong, Y., 2006. Product configuration optimization using a multiobjective GA. International Journal of Advanced Manufacturing Technology 30, 20–29.

Michalski, R.S.: Learnable Evolution: Combining Symbolic and Evolutionary Learning, Proceedings of the Fourth International Workshop on Multistrategy Learning (MSL’98), p. 14–20, 1998.

Michalski, R.S., Wojtusiak, J., Kaufman, K.A.: Intelligent Optimization via Learnable Evolution Model, 18th IEEE Conference on Tools with Artificial Intelligence, p332–335, (2006).

Paris, J.L., Tautou-guillume, L., Pierreval, H., 2001. Dealing with design options in the optimization of manufacturing systems: an evolutionary approach. International Journal of Production Research 39 (6), 1081–1094.

Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Francisco.

Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: Linkage problem, distribution estimation, and Bayesian networks, IlliGAL Report No. 98013, Illinois Genetic Algorithm Laboratory, University of Illinois, Urbana, 1998.

Rochet, S., Baron, C.: An Evolutionary Algorithm for Decisional Assistance to Project Management, in Handbook of Research on Nature Inspired Computing for Economy and Management, vol. 2, section 2.1, 2006.

Schwarz, J., Ocenasek, J., 2000. A problem knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning, Fourth Joint Conference on Knowledge-Based Software Engineering. IOS Press, p. 51–58.

Sebag, M., Schoenauer, M., 1994. A rule based similarity measure. Topics in Case-Based Reasoning, vol. 837. Springer-Verlag, p. 119–130.

Sebag, M., Ravise, C.: An Advanced Evolution Should Not Repeat Its Past Errors, Proceedings of the 13th International Conference on Machine Learning, 1996.

Smith, J., Fogarty, T.C.: Recombination strategy adaptation via evolution of gene linkage, In: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, p. 826–831, (1996).

Tanner, M., 1996. Tools for Statistical Inference Third Edition Springer Verlag, New York.

Zitzler, E., Thiele, L., 1999. Multi objective EA: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3 (4), 257–271.